
Magento Performance Testing Guidelines

TABLE OF CONTENTS

OVERVIEW	2
Methodology	2
Objectives	2
Terminology	2
Recommended Test Protocol	3
CLOUD SERVICES VS. DEDICATED HOSTING.....	4
SYSTEM RESOURCE MONITORING.....	4
MAGENTO CONFIGURATION SETTINGS	5
Use Flat Catalog	5
Merge JavaScript & CSS.....	5
Enable Full-Page Cache.....	6
Magento Indexers.....	7
SERVER CONFIGURATION SETTINGS.....	8
.htaccess.....	8
Temporary File Storage	8
Cron Job	9
LAMP STACK.....	9
Linux	9
Apache.....	11
MySQL	11
PHP	12
LNMP STACK.....	13
Linux	13
Nginx	13
MySQL	14
PHP	14

- UNSUPPORTED CONFIGURATIONS AND SETTINGS 15**
 - Linux 15
 - Apache..... 15
 - MySQL 15
 - PHP 16
- SAMPLE CONFIGURATIONS 17**
 - MySQL Configuration 17
 - Nginx Configuration 19
- EXAMPLE SYSTEM CONFIGURATIONS 20**
 - System Hardware Specifications 20
 - Physical Servers 20
 - Cloud Servers 20
 - System Software Versions 21
 - Developer Test Environments 21
- SERVER TEST ENVIRONMENTS (1 SERVER) 22**
- SERVER TEST ENVIRONMENTS (2 SERVERS) 23**
- MAGENTO SCALABILITY 24**
 - Magento Scalability Tiers..... 24
 - Performance vs. Scalability 24

EXECUTIVE SUMMARY

Performance testing is a critical step in the development process for merchants looking to optimize their shopping experiences.

This technical document provides a foundation for performance testing by defining a common set of software and environment configuration settings to produce reliable, reproducible, and consistent performance test results.

These *Guidelines* are meant to be used in conjunction with the Magento Performance Toolkit, and together they provide a consistent ruler to compare performance across a variety of Magento implementations.

Using the *Guidelines* and Toolkit to gather performance measurements across different implementations provides merchants or their developers with information on the performance impacts of custom hardware, software, or network configurations, empowering them to optimize their web store experience for growth.

OVERVIEW

Methodology

1. Load a Magento instance on a server, or set of servers.
2. Use the [Magento Performance Toolkit](#) to prepare the database for testing.
3. Use [Apache jMeter](#) to run benchmarking scripts from a load generator to send traffic to the server(s) and collect performance data.

Objectives

- Enable merchants to test the performance of Magento installations
- Enable developers to test the performance impact of customizations
- Enable performance testing across the Magento ecosystem
- Enable the comparison of performance tests (benchmarking)
- Provide commonly used methods, scenarios, and terminology to describe testing activities

Terminology

The following terms have specific meaning within the context of Magento performance testing:

Performance	Responsiveness and stability under a particular workload.
Benchmarking	Comparing system performance metrics, typically to recommended best practices or industry standards.
Scalability	<ol style="list-style-type: none">1. The ability to process growing amounts of requests in a capable manner.2. The ability to expand the system resources to increase capacity.
Reliability	<ol style="list-style-type: none">1. The ability to yield the same or compatible results for different tests.2. The ability for the system to operate as designed.
Reproducibility	<ol style="list-style-type: none">1. The ability to conduct a performance test and observe similar results on multiple test occurrences.2. The ability for a third party to independently duplicate a test and observe comparable results.

Recommended Test Protocol

Document the environment and test procedures so the results are comparable and reproducible.

Record all control variables.

Performance test control variables are divided into the following groups:

- Load Generator
- Merchant Traffic Profile
- Reference Store
- Magento Platform
- Environment
- Hardware
- Software
- Reference System Architecture
- Test Protocol

Minimize test variables.

Minimize test variables. When testing to compare results, minimize the number of input variables that are subject to change.

Drive tests from different servers.

To isolate system resource usage, place the load generator on a different system than the application server that runs Magento and supporting services. Large test configurations might need multiple load generators to create enough traffic to stress the system.

Establish a baseline.

Establish a baseline set of results with known values for the full list of control variables. The recommended values for several control variables are described later in this document.

Don't use the first set of results.

For cached configurations, the first test cycle populates only the caching layers. For each cycle, record the results of the second and subsequent test runs.

Repeat to verify results.

Never consider the first set of results as final. Repeat and verify each test to establish that the results are reliable and reproducible by comparing the variations in results for each test cycle. A minimum of two or three test cycles is recommended.

Include the minimum number of external variables in your test environment.

The network metric distance from the load generator to the application server should be the minimum distance that is required to establish the scope of the test. Your intent should be to minimize the impact of external variables on the test results, and know which variables are impacting your results.

- A test from a load generator to a server on the same switch can be used to establish the scope of application performance and hardware configuration.
- A test from an Internet source to a private data center can be used to establish the scope of Internet latency, CDN and caching servers, infrastructure equipment, application performance, and hardware configuration.

CLOUD SERVICES VS. DEDICATED HOSTING

Magento was originally designed to be installed and run on a dedicated, physical server, and deployed as an on premise application. The emerging trend is to install Magento in a data center on either a dedicated physical server, or using a cloud service provided by the host.

Servers that are managed by a hosting provider and provisioned for multiple customers are referred to as **cloud services**. A physical server that is used by a single customer is referred to as a **dedicated host**.

SYSTEM RESOURCE MONITORING

It is recommended that you collect system-level metrics during the execution of the benchmark, to make sure the results are not based on a system resource that is over utilized or limited causing a resource bottleneck. At a minimum, monitor the usage for each of the following primary system resources:

- CPU utilization
- Memory allocation
- Disk IO activity
- Network usage
- Database transactions

To increase your understanding of how these resources are used you want to collect detailed usage data during your test activities. You can use various tools to collect these metrics including jMeter test plan listeners, individual system level commands, and system monitoring tools such as nmon. Resource usage results over several test runs can be captured and visualized in tools like Graphite. Also, you may consider using SaaS monitoring tools like New Relic to capture and visualize data.

MAGENTO CONFIGURATION SETTINGS

The following configuration settings are required for each Magento store to be tested:

- [Use Flat Catalog](#)
- [Merge JavaScript & CSS](#)
- [Enable Full-Page Cache](#)
- [Indexers Update and Clean on Schedule](#)

Use Flat Catalog

To configure Magento to enable Flat Catalog for categories and products, do the following:

1. On the Admin menu, select **System > Configuration**. In the panel on the left, under Catalog, select **Catalog**.
2. Click to expand the **Frontend** section. Then, make the following settings:

Use Flat Catalog Category "Yes"

Use Flat Catalog Product "Yes"

Merge JavaScript & CSS

To enable file merging for JavaScript and CSS files, do the following:

1. In the panel on the left, under Advanced, select **Developer**.
2. Click to expand the **JavaScript Settings** section. Then, set **Merge JavaScript Files** to "Yes."
3. Click to expand the **CSS Settings** section. Then, set **Merge CSS Files** to "Yes."
4. When complete, click the **Save Config** button.

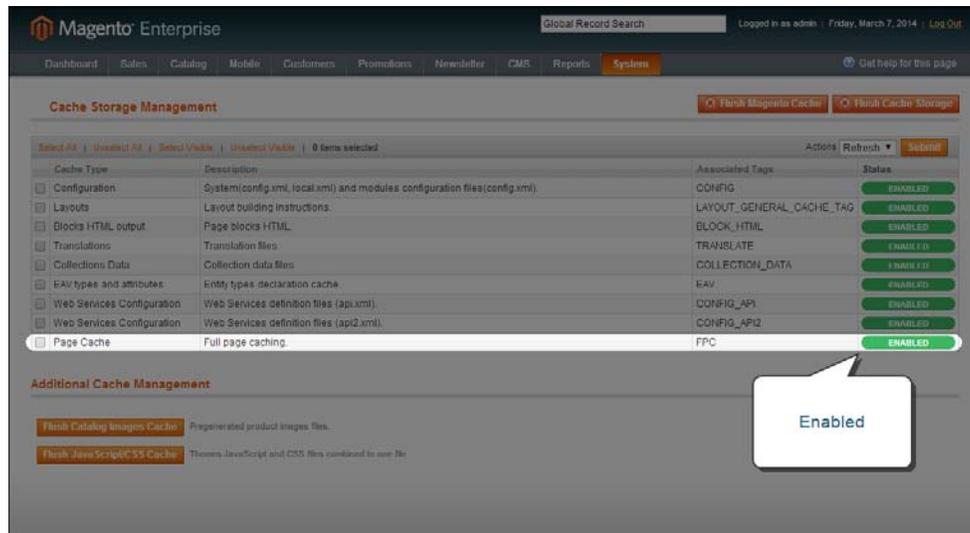
Enable Full-Page Cache

For Magento Enterprise Edition:

The Full-Page Cache is enabled by default for Magento Enterprise Edition. To verify that the page cache is enabled, do the following:

Step 1: Verify that Page Cache is Enabled

1. On the Admin menu, select **System > Cache Management**.



2. If the **Status** column shows the Page Cache to be “Disabled,” do the following:
 - a. On the Admin menu, select **System > Configuration**. In the panel on the left, under Advanced, select **System**.
 - b. Click to expand the **Full Page Cache Settings** section. Then, set **Enable Full Page Cache Auto Generation** to “Yes.”
3. When complete, click the **Save Config** button.

Step 2: Verify that Page Cache is Valid

On the server, examine the contents of the cache to verify that it is working. It is not possible to determine the location of the cache from the Magento Admin. Use one of the following methods to verify the cache:

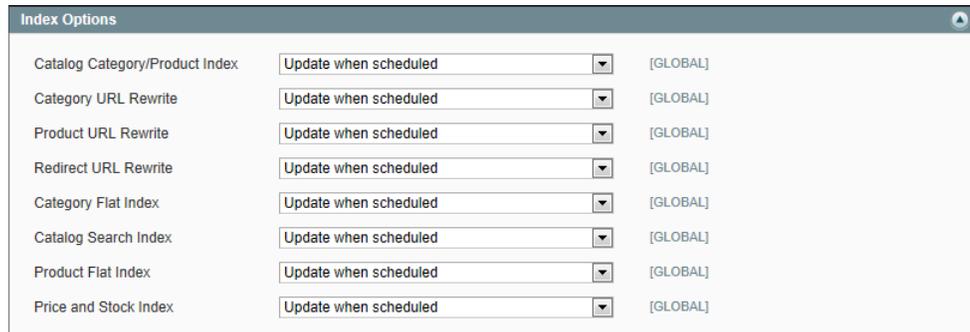
- File System If using the file system, verify that cached pages are being saved in the following folder:
<magento root>/var/full_page_cache
- Redis If using Redis, make an API call to query data used by Magento.

Magento Indexers

For Magento Enterprise Edition:

All indexers should “Update when scheduled,” and the system should periodically purge the indexer changelog.

1. On the Admin menu, select **System > Configuration**. In the panel on the left, under Advanced, select **Index Management**.
2. Click to expand the **Index Options** section. Then, verify that all indexers are set to “Update when scheduled.”



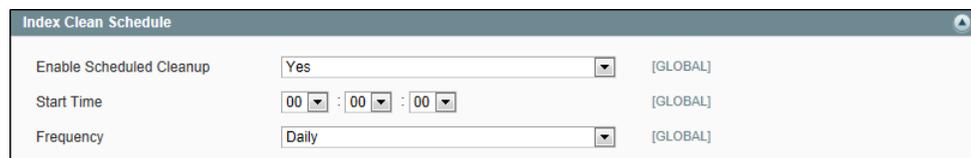
Index Options		
Catalog Category/Product Index	Update when scheduled	[GLOBAL]
Category URL Rewrite	Update when scheduled	[GLOBAL]
Product URL Rewrite	Update when scheduled	[GLOBAL]
Redirect URL Rewrite	Update when scheduled	[GLOBAL]
Category Flat Index	Update when scheduled	[GLOBAL]
Catalog Search Index	Update when scheduled	[GLOBAL]
Product Flat Index	Update when scheduled	[GLOBAL]
Price and Stock Index	Update when scheduled	[GLOBAL]

3. Click to expand the **Index Clean Schedule** section. Then, verify the following settings:

Enable Scheduled Cleanup “Yes”

Start Time (As needed)

Frequency (As needed)



Index Clean Schedule		
Enable Scheduled Cleanup	Yes	[GLOBAL]
Start Time	00 : 00 : 00	[GLOBAL]
Frequency	Daily	[GLOBAL]

4. When complete, click the **Save Config** button.

SERVER CONFIGURATION SETTINGS

.htaccess

Rewrite Engine = ON

In vi .htaccess, make sure that this line is not comment: #RewriteEngine On

This is the default setting for the htaccess file included with Magento.

```
RewriteEngine On
```

Expire Headers = ON

The correct Expire Headers setting can be found in the default Magento htaccess file.

The default setting is access time +1 year.

Temporary File Storage

Configure the location of the following temporary files as needed for your system architecture:

```
<magento root>/var/session  
<magento root>/var/cache  
<magento root>/var/full_page_cache
```

SYSTEM TYPE	STORAGE LOCATION
File System	Store all caches and session in memory backed by file system, or store on a RAM disk.
Redis	Store all caches and session in Redis on the application server.
1+1	Store all caches and session in database.
1+1 with Redis	On Application Server: Store in Redis on the application server. On Dedicated Server: Store in Redis on the Redis server instance.
2 + 1	Store all caches and session in database.
2 + 1 with Redis	Store all caches and session in Redis on the Redis server instance. In this example, the load balancer can be either hardware or a server instance.

Cron Job

For Magento Enterprise 1.12:

The Magento Cron job should be scheduled to run at intervals between five and ten minutes. Schedule the interval to run no sooner than five minutes, and no later than ten minutes.

For Magento Enterprise 1.13 or later:

The Magento Cron job should be scheduled to run at one minute intervals, and be executed by the specified user. It is recommended to run the Cron job under Apache rather than root. The same user should run Cron, Magento, and PHP. To learn more, see: [How to Set Up a Cron Job](#)

1. From the command line, enter the following command to run crontab:

```
crontab -u apache -e
```

2. Add the following line:

```
[Magento 1.12]

*/5**** /bin/sh/absolute/path/to/magento/cron.sh

[Magento 1.13+]

*/1**** /bin/sh/absolute/path/to/magento/cron.sh
```

LAMP STACK

Linux

Optimize disk performance:

1. Mount the disk volume with `noatime` and `nodiratime`.
2. Add one of the following lines to `vi /etc/fstab`:

For ext3 file system:

```
/dev/md0 / ext3 defaults,noatime,nodiratime,barrier=0 1 1
```

For ext4 file system:

```
/dev/md0 / ext4 defaults,noatime,nodiratime,barrier=0 1 1
```

Set no write barriers:

For benchmarking, write barriers should always be set to zero. For a production environment, your system administrator can enable or disable write barriers, depending on the hardware. Under certain conditions, an incorrect setting of this parameter can cause data loss.

```
barrier=0
```

Set swappiness threshold:

Set the swappiness threshold to ten or lower. Do not adjust up if the value is less than ten, including zero.

```
/proc/sys/vm/swappiness
```

Increase max file descriptors:

Set value to 65535, default value 1024

Disable transparent huge page support:

Disable Transparent Huge Page Support on the host OS.

Enabling THP on the Host OS has shown modest performance improvements on guest machines. However, after several weeks in production on a busy system with substantial memory (128GB), guest machines can become non-responsive for an extended period of time (6,000ms+) during memory defragmentation.

```
thp_enabled = false
```

Enable reuse of TIME-WAIT sockets:

Enable TCP_TW_REUSE so that the sockets in TIME_WAIT state can be used again instead of having to create new sockets for every new request.

```
net.ipv4.tcp_tw_reuse = 1
```

Apache

- URL Rewrites = enabled
- Sudo a2enmod rewrite

Apache Multi-Process Modules (MPM):

When PHP-FPM is not used

```
MPM = prefork
```

When PHP-FPM is used

```
MPM = worker
```

When PHP-FPM is used with Apache 2.4

```
MPM = event  
  
maxclients = 256
```

MaxClients:

Sets the maximum number of server processes that are allowed to start. (This parameter is dependent on MPM settings.) For more information, see: [How to Set MaxClients in Apache/prefork](#).

MySQL

- Version: MySQL/Percona 5.6 or later

The Magento 1.12 installer uses a MySQL variable to check for innodb support. This variable was deprecated in MySQL 5.6. For more information, see: [MySQL Configuration](#).

Set buffer pool size:

Adjust the MySQL buffer pool size according to the type of server.

Multi-purpose Server	40%
----------------------	-----

Dedicated Database Server	75%
---------------------------	-----

```
innodb_buffer_pool_size
```

Set maximum connections:

For testing, set the maximum connections to 2,000. For production servers, it is recommended that you double the number of expected concurrent sessions. (2* concurrent user sessions.)

```
max-connections=2000
```

Set thread concurrency:

When using current versions of MySQL (5.6), the recommended value for this setting is 16. If your server has several CPU cores available, it can be changed to a value between 16–32.

```
innodb-thread-concurrency = 16
```

PHP

- Version: PHP 5.4.x or PHP 5.5.x

Enable OPcache:

Load and enable the OPcache module.

Set PHP memory limit:

Edit the PHP memory limit in php.ini.

The Magento Performance Toolkit includes minimum required values for PHP memory limit, based on profile size.

For a 4GB application server, using the medium profile, the minimum PHP memory limit should be 1G.

Set PHP OPcache memory limit:

Edit the PHP OPcache memory limit in php.ini. The OPcache memory limit must be smaller than the PHP max memory limit.

```
[opcache]
opcache.enable=1
opcache.enable_cli=1
opcache.memory_consumption=256
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=4000
opcache.max_wasted_percentage=5
opcache.use_cwd=1
opcache.validate_timestamps=1
opcache.fast_shutdown=1
```

Disable APC:

Disable or remove the Alternate PHP Cache (APC)

Disable Xdebug:

Do not use Xdebug for benchmarking or for production systems.

Turn off PHP error display:

Edit php.ini as follows:

```
display_errors = off
```

Limit PHP error reporting for production systems:

This change is appropriate when you don't want to display notices or other non-fatal errors on a production system. Edit php.ini as follows:

```
error_reporting = E_WARNING | E_ERROR  
  
log_errors = on  
  
error_log = /path/store/error/log/php_errors.log
```

Set garbage collection probability

This value has a default of 1, and is recommended for benchmarking, development, and production systems. Edit php.ini as follows:

```
session.gc_probability = 1
```

LNMP STACK

The following examples illustrate the use of control variables for the software environment on a LNMP stack system.

Linux

(No change from LAMP settings.)

Nginx

Use Nginx and FastCGI (PHP-FPM):

1. Set up Nginx.

```
apt-get install Nginx  
  
apt-get install php5-fpm
```

2. Configure PHP5-FPM to listen on socket 9000.
 - Set up php-fpm & nginx config for php as appropriate.
 - Set up nginx to cache/serve static assets (img,js).
 - Set up nginx to set expiration headers for static files.
3. Set static cache.

For more information about the official Magento configuration, see: [Configuring Nginx for Magento](#). Look online for php5-fpm set up and configuration instructions.

Use Unix sockets:

When using Nginx and a single server configuration, you can use Unix sockets to reduce TCP traffic and latency.

```
upstream fastcgi { server unix:/var/run/php-fpm.sock; }
```

Tune Nginx worker:

Tune Nginx worker processes and worker connections. These values can be tuned to find an optimal configuration, depending on your server environment. For the purposes of testing they should be 1 and 1024.

```
worker_processes = 1    # adjust to equal the number of CPUs
worker_connections = 1024
```

For more information, see [Nginx Configuration](#) in the Sample Configurations section of this document.

MySQL

(No change from LAMP settings.)

PHP

New wire mode for PHP-FPM:

Configure PHP-FPM to listen on Unix socket mode.

```
listen = /var/run/php-fpm.sock
```

UNSUPPORTED CONFIGURATIONS AND SETTINGS

The following settings should not be used for benchmarking.

These are provided as examples of settings that you should not change for the purpose of performance testing. You can, at your own discretion, use these settings for the purposes of analyzing performance and tuning your own development system or changing system behavior on production servers. They have been proven to effect results in either a positive or negative way, but shouldn't be used for benchmarking.

Linux

Use ramdisk for var/cache and var/session files:

For a system without a key storage, do the following:

1. Create and mount ramdisk.

```
mkdir /mnt/ramdisk  
mount -t tmpfs -o size=512m tmpfs /mnt/ramdisk
```

2. Add the following line to /etc/fstab:

```
tmpfs /mnt/ramdisk tmpfs nodev,nosuid,noexec,nodiratime,size=512M 0 0
```

3. Set up /tmp to use ramdisk along with Magento's var/cache and var/session.

```
-s MAGENTOROOT/var/cache/tmp/magentocache  
-s MAGENTOROOT/var/session/tmp/magentosession
```

Apache

Use vhost directive instead of directory .htaccess files:

You can move the contents of the Magento .htaccess file into apache vhost directive. This change can have a noticeable impact on file system activity, and may show positive results in production server environments.

MySQL

Do not use SQL profiling:

Profiling can be enabled when looking for queries to tune, but should be disabled when conducting performance benchmarks.

PHP

Increase max execution time:

Increase the PHP max execution time for web admin servers. This value is hard coded to 0 (unlimited) for the PHP CLI. However, on a Magento admin web server, it is possible to run into a timeout condition if this value is not set high enough to accomplish the admin tasks requested (e.g. a large import). For a customer-facing production web server, setting this value too high may allow the server to over-allocate resources.

Edit php.ini as follows:

Production Application Server: (default setting)

```
max_execution_time = 30
```

Admin Web Server

```
max_execution_time = 5000
```

Turn off OPcache timestamp checking:

Turning off timestamp checking will give some performance gain, but it requires you to manually flush the cache anytime you modify a file. Modifying files and failing to flush that cache will result in server errors.

```
[opcache]

opcache.validate_timestamps=0
```

SAMPLE CONFIGURATIONS

MySQL Configuration

The following MySQL parameters are recommended for performance testing only. For a production server, a qualified database administrator can provide the best configuration settings for your environment. The best settings can vary depending on your database server capacity, size of your store, and/or website traffic characteristics.

MySQL Configuration for SMALL or MEDIUM Profile:

```
[mysqld]
datadir=/var/lib/mysql
#socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
#symbolic-links=0

max_connections = 1000
max_connect_errors = 10000000
max_allowed_packet = 32M
max_heap_table_size = 64M
sort_buffer_size = 8M
join_buffer_size = 128K
thread_cache_size = 16
query_cache_size = 32M
query_cache_limit = 2M
tmp_table_size = 64M # (same size as Heap Table)
key_buffer_size = 32M
read_buffer_size = 8M
read_rnd_buffer_size = 8M
bulk_insert_buffer_size = 64M
wait_timeout = 28800 # (default value)
innodb_additional_mem_pool_size = 16M
innodb_log_buffer_size = 8M
innodb_log_file_size = 256M
innodb_log_files_in_group = 2
innodb_buffer_pool_size = 6G
                #(6G is appropriate, based on 7-8GB system memory for database server)
innodb_data_home_dir = /var/lib/mysql/
innodb_data_file_path = ibdata1:256M;ibdata2:256M:autoextend
innodb_autoextend_increment = 256
innodb_thread_concurrency = 16

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

MySQL Configuration for LARGE or Extra Large Profile:

```
[mysqld]
datadir=/var/lib/mysql
#socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
#symbolic-links = 0

max_connections = 1000
max_connect_errors = 10000000
max_allowed_packet = 32M
max_heap_table_size = 128M
sort_buffer_size = 8M
join_buffer_size = 128K
thread_cache_size = 16
query_cache_size = 32M
query_cache_limit = 2M
tmp_table_size = 128M # (same size as Heap Table)
key_buffer_size = 32M
read_buffer_size = 8M
read_rnd_buffer_size = 8M
bulk_insert_buffer_size = 64M
wait_timeout = 28800 (default value)
innodb_additional_mem_pool_size = 16M
innodb_log_buffer_size = 8M
innodb_log_file_size = 512M
innodb_log_files_in_group = 2
innodb_buffer_pool_size = 6G
# (6G is appropriate, based on 7-8GB system memory for database server)
innodb_data_home_dir = /var/lib/mysql/
innodb_data_file_path = ibdata1:1G;ibdata2:512M:autoextend
innodb_autoextend_increment=512
innodb_thread_concurrency = 16

[mysqld_safe]
log-error=/var/log/mysql.log
pid-file=/var/run/mysql/mysql.pid
```

Nginx Configuration

Magento recommends this sample configuration for testing Nginx.

Nginx Test Configuration

```
sendfile on;           # enable operating system buffering
server_tokens off;     # do not report nginx version in headers
tcp_nopush on;        # do not wait for TCP push socket parameter
keepalive_timeout 10; # limit the length of a connection timeout, reduce thread creation
open_file_cache max=10000 inactive=30m; # enable caching of open file descriptors
open_file_cache_valid 10m;
open_file_cache_min_uses 2;
open_file_cache_errors on;
gzip on;               # enable compression of content
gzip_disable "msie6";
gzip_vary on;
gzip_proxied any;
gzip_comp_level 5;
gzip_buffers 16 8k;
gzip_http_version 1.1;
gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml
application/xml+rss text/javascript; # specify content types
```

EXAMPLE SYSTEM CONFIGURATIONS

System Hardware Specifications

These recommendations are for consistent testing purposes only. Production servers can have a variety of optimized configurations. The minimum and recommended test system configurations are:

Physical Servers

PHYSICAL SERVER	MINIMUM	RECOMMENDED
Web Application Server	4 CPU Cores	8 CPU Cores
	8 GB RAM	16 GB RAM
Database Server	4 CPU Cores	8 CPU Cores
	8 GB RAM	16 GB RAM
	Direct Attached Storage	
Auxiliary Server (Redis, Memcached, etc.)	1 CPU	Multi-Core CPU
	2 GB RAM	4 GB RAM
Load Generator*	4 CPU Cores	8 CPU Cores
	8 GB RAM	16 GB RAM

Cloud Servers

PHYSICAL SERVER	MINIMUM	RECOMMENDED
Web Application Server	4x vCPU	8x vCPU
	8 GB	16 GB
	Cloud Storage	
Database Server	4x vCPU	8x vCPU
	8 GB	16 GB
	Cloud Storage	Preferred Storage
Auxiliary Server (Redis, Memcached, etc.)	1x vCPU	1x vCPU
	2 GB RAM	4 GB RAM
	Cloud Storage	
Load Generator*	4x vCPU	8x vCPU
	8 GB	16 GB

* The Load Generator becomes the performance bottleneck at a certain traffic levels. It is recommended that you scale up the number of load generators instead of adding resources to a single load generator.

System Software Versions

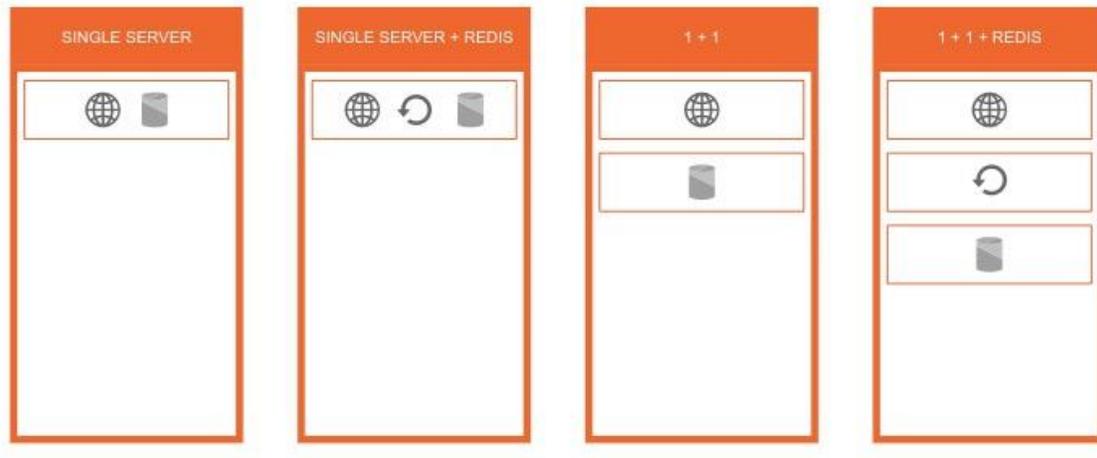
Magento EE v1.12+	The minimum requirements in this document are based on Magento Enterprise Edition, version 1.12. Similar configuration options should be used for later versions.
PHP 5.4, PHP 5.5	The PHP patch should be applied as necessary to confirm that the Magento system as applied.
Zend OPcache	For PHP 5.4, OPcache can be loaded optionally For PHP 5.5, Zend OPcache is included by default

Developer Test Environments

A developer who is testing code needs an environment that minimizes the hardware impact on test results, in order to focus on the impact of the software environment, software settings, and code changes. The system should have a fast CPU, sufficient RAM, and drive storage that is fast, affordable, and that can produce consistent results.

Developer Performance System	<ul style="list-style-type: none">• Quad-Core CPU• 32GB RAM• SSD HDD
------------------------------	--

SERVER TEST ENVIRONMENTS (1 SERVER)

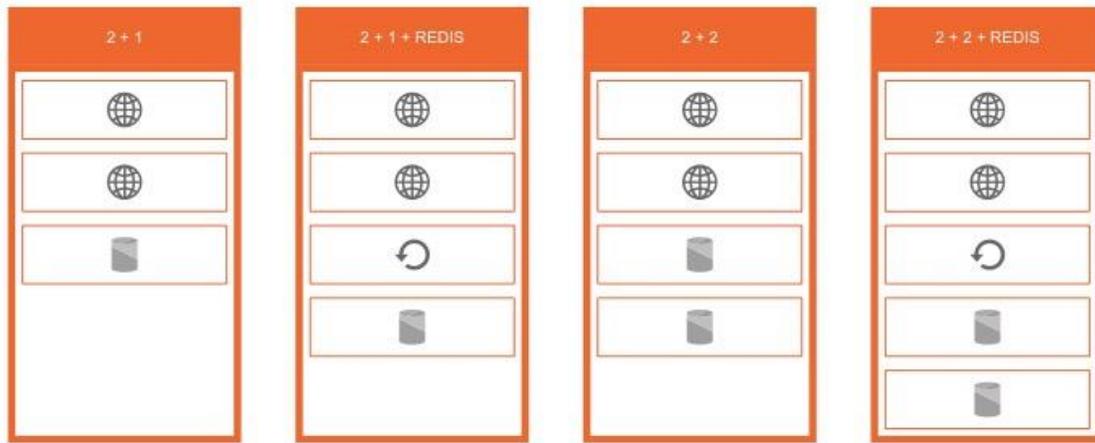


A single server should be sufficient to test a SMALL or MEDIUM profile.

SINGLE SERVER ENVIRONMENT

Single Server	One server that is both web application server and database server.
Single Server + Redis	One server that is both web application server and database server. The Redis cache is also used on the same server.
1 + 1	One web application server and one database server.
1 + 1 + Redis	One web application server, one database server, and one Redis server.

SERVER TEST ENVIRONMENTS (2 SERVERS)

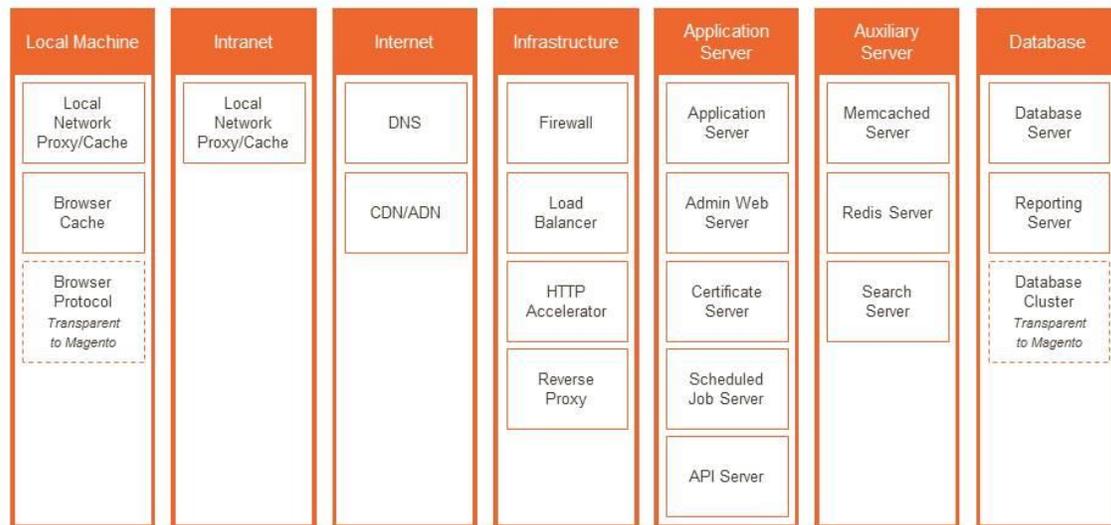


A minimum of two web application servers is recommended to test traffic with a LARGE or EXTRA LARGE profile.

TWO-SERVER ENVIRONMENT

2 + 1	Two web application servers, and one database server.
2 + 1 + Redis	Two web application servers, one database server, and one Redis server.
2 + 2	Two web application servers, and two database servers. (The recommended settings for this configuration are not included in this document.)
2 + 2 + Redis	Two web application servers, two database servers, and one Redis server. (The recommended settings for this configuration are not included in this document.)

MAGENTO SCALABILITY



Magento Scalability Tiers

An example set of variables for scalability testing are shown in the diagram above. These would be independent variables in a fixed configuration environment, and some could be dependant variables in an auto-scaling environment.

Performance vs. Scalability

Magento performance testing is related to scalability, but they are not the same.

You can measure the performance of a particular configuration of your system. You can then either (a) change a variable in your environment and test again to measure performance impact, or (b) change the traffic level against the same environment and test scalability.

- Verifying that 1,000 user sessions can hit your Magento store and that the average response time of 1,000ms or faster is performance testing.
- Changing your software or hardware configuration to reduce that number from 1,000ms to 500ms is a performance improvement.
- Increasing the user session count from 1,000 to 10,000 and confirming that the average response time is still 1,000ms or faster is scalability testing.